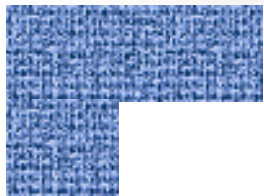


Object Relations and Syntactic Mechanisms in Design Patterns



Uriel Cohen

Introduction

■ Q: How can we capture design knowledge in software engineering?

■ A: *Design patterns*

■ Elements of a design pattern:

- *name*
- *problem*
- *solution*
- *consequences*



```
#include <iostream.h>
int main()
{
    int i;
    (i=0;i<10;i++)
        cout << i;
    ...
    ...
    return 0;
}
```





Design Patterns

■ Advantages:

- Common language
- Higher level of abstraction
- Code-reusability

■ Problems:

- *Traceability problem*
- *Reusability problem*
- *Implementation overhead*





Motivation of the Research

- Design patterns as lingual features.
- Bosch, 1998 – LayOM.
- Agerbo and Cornils, 1998 – class library of *Fundamental Design Patterns* in BETA.
- Chambers, Harrison and Vlissides, POPL 2000 – A debate on language and tool support for design patterns.



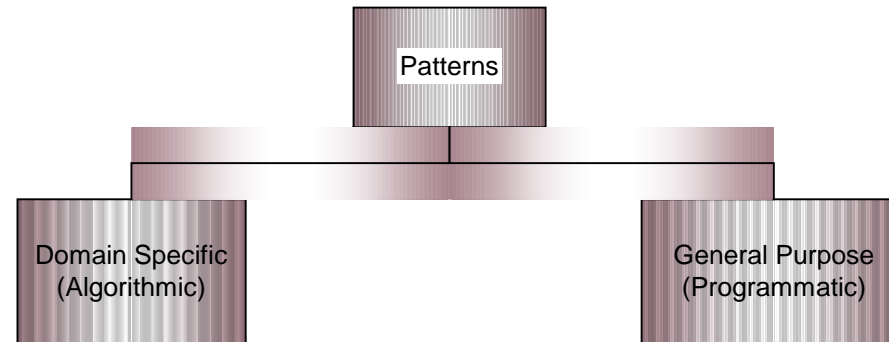


Course of Action

1. Data set of design patterns.
2. Investigate:
 - Syntactic mechanisms in design patterns.
 - Objects relations in design patterns.
3. Propose useful lingual features.



Our Taxonomy of Patterns

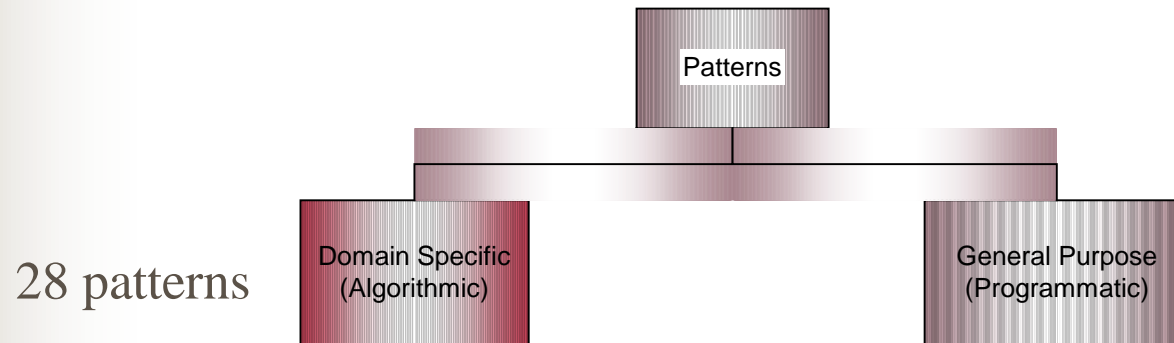


- A lot of research involved.
- Differentiates between patterns and design patterns.
- Allows us to select a data set of patterns.

Source of DPs: Tichy, 1997

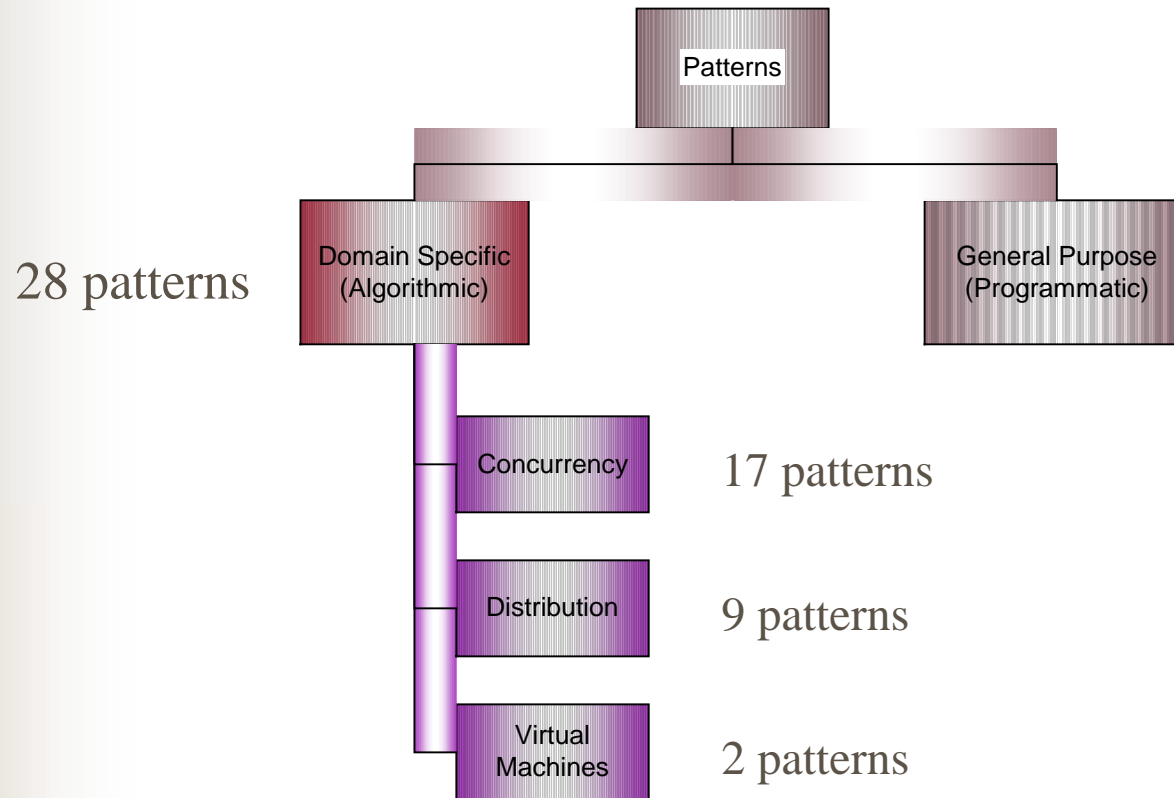


Our Taxonomy of Patterns



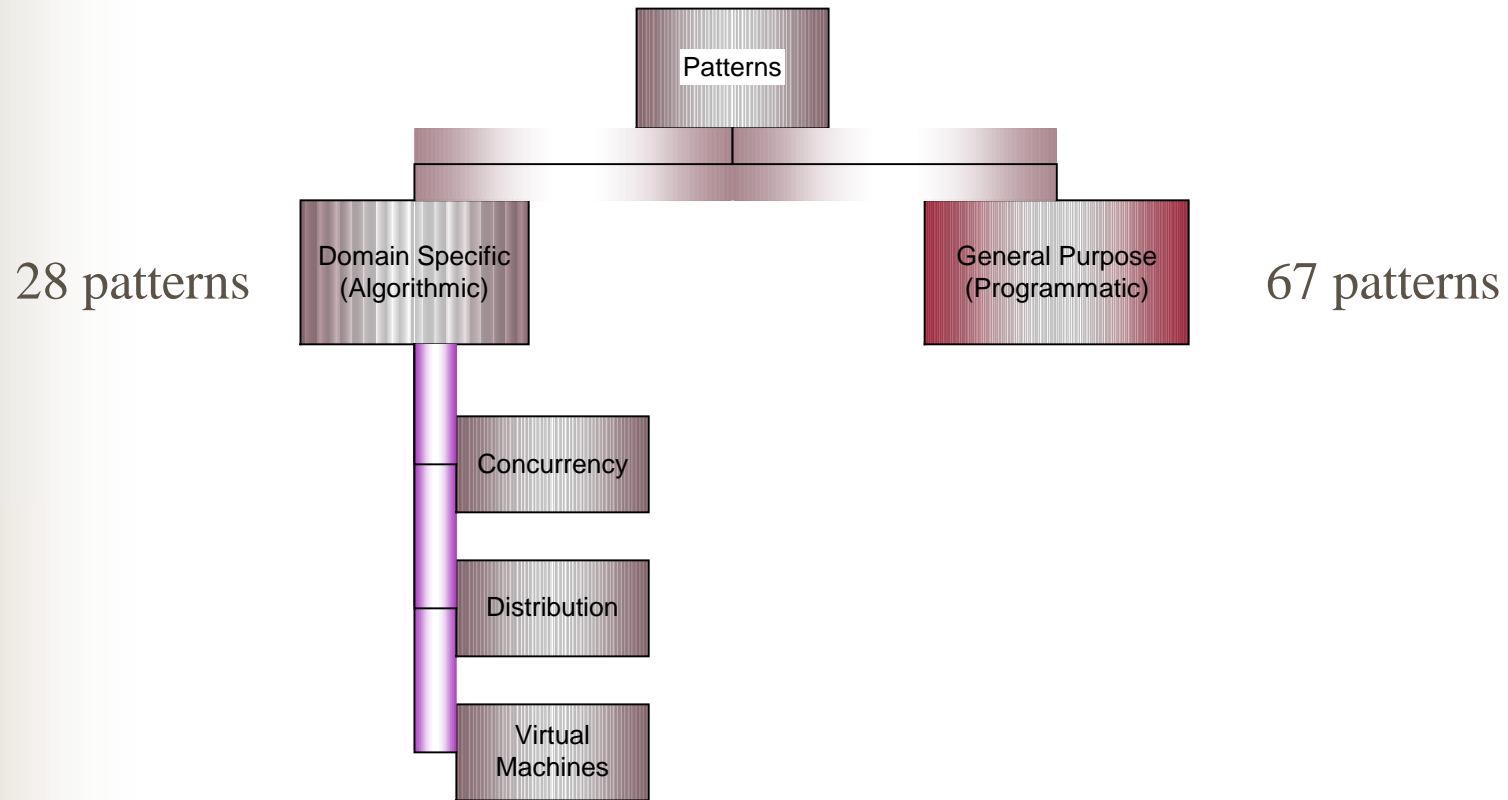
Source of DPs: Tichy, 1997

Our Taxonomy of Patterns



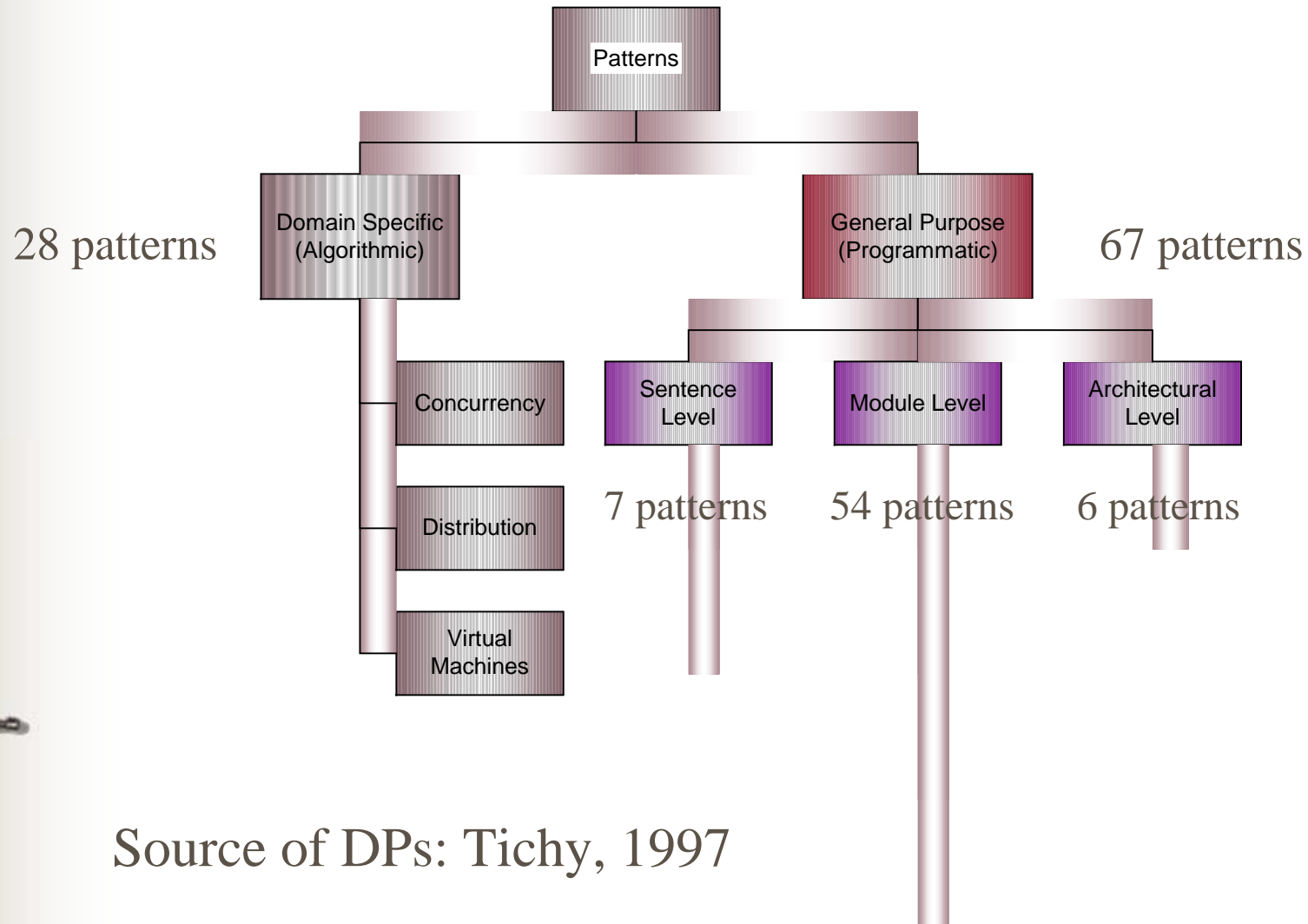
Source of DPs: Tichy, 1997

Our Taxonomy of Patterns



Source of DPs: Tichy, 1997

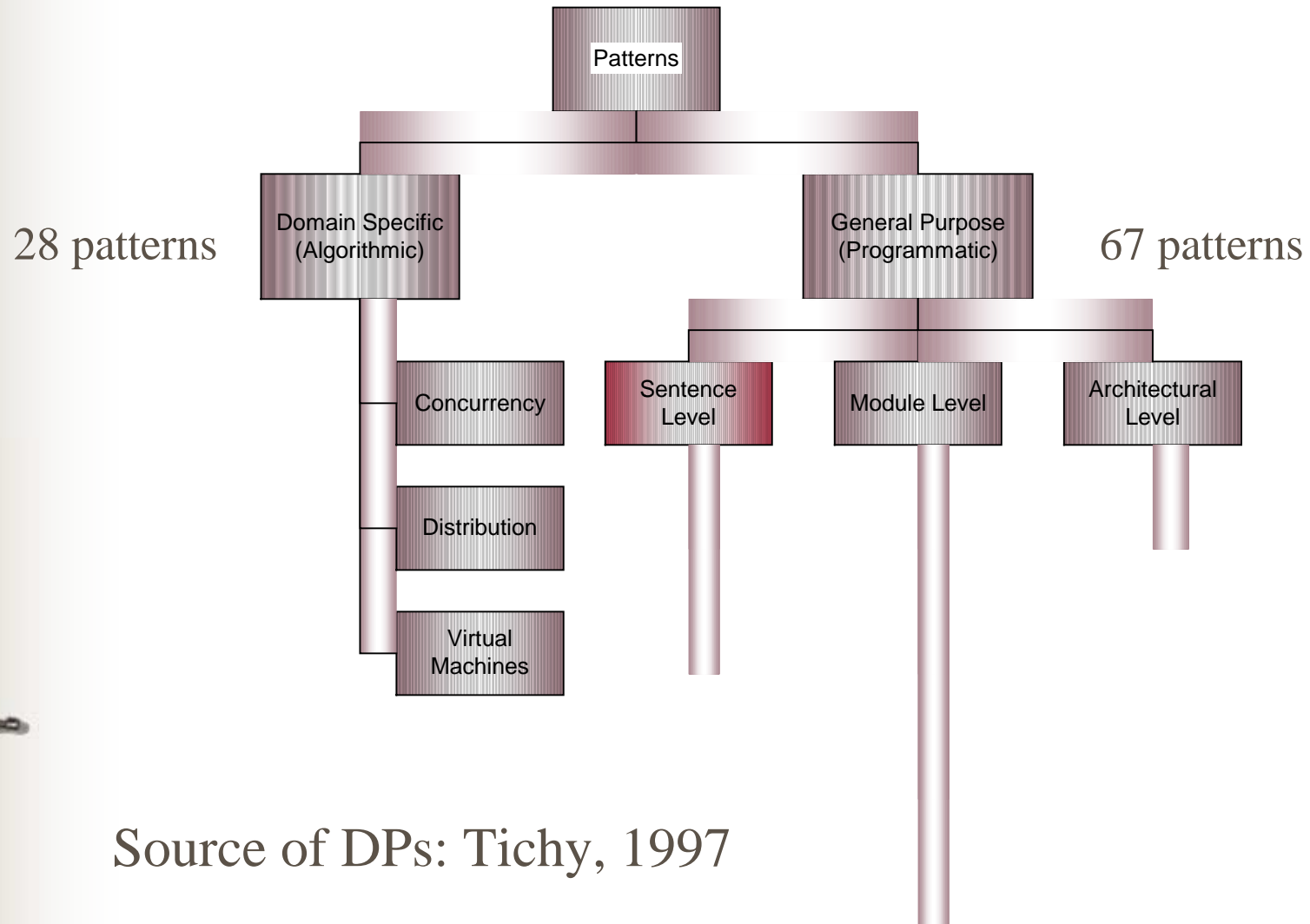
Our Taxonomy of Patterns



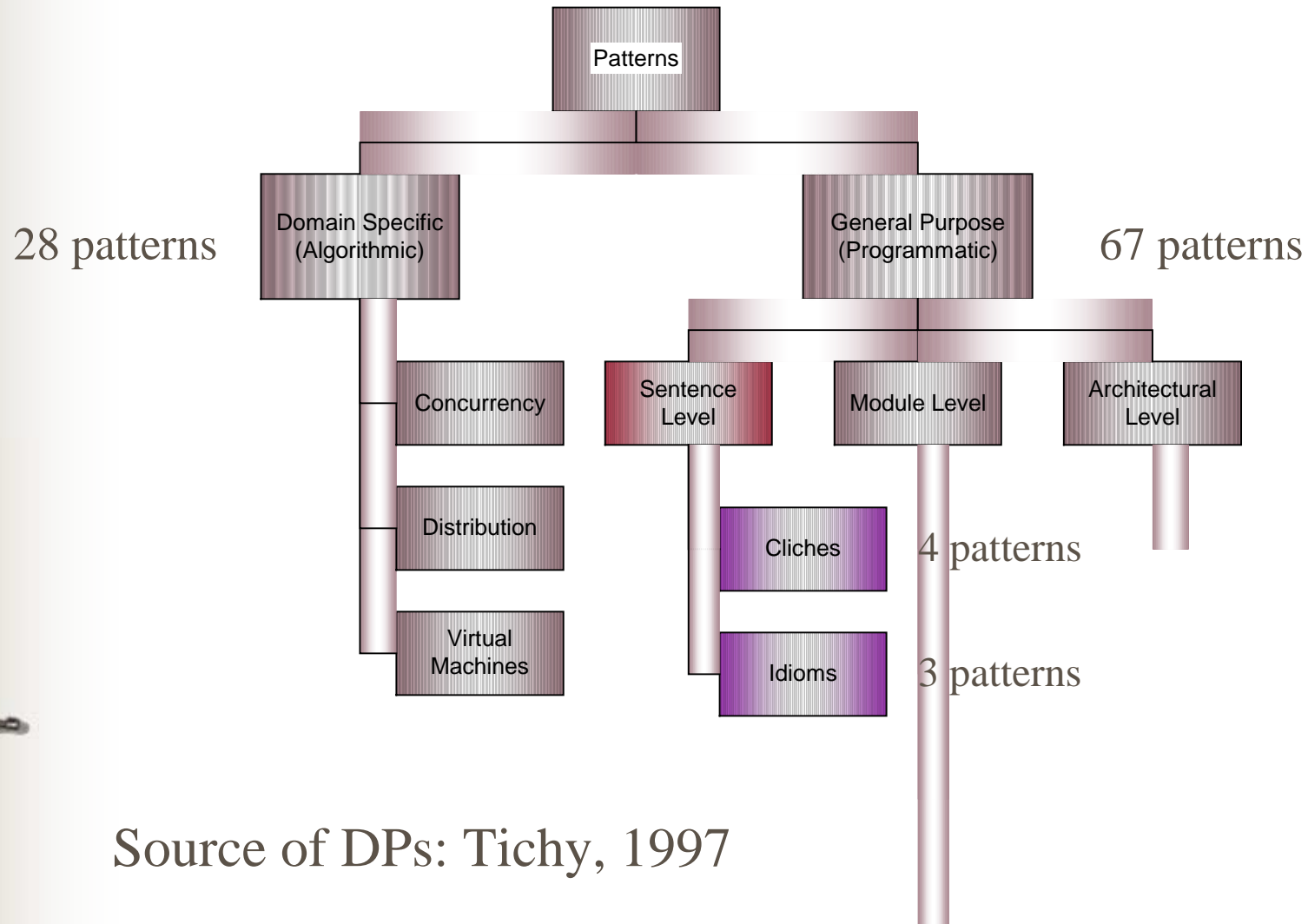
Source of DPs: Tichy, 1997



Our Taxonomy of Patterns



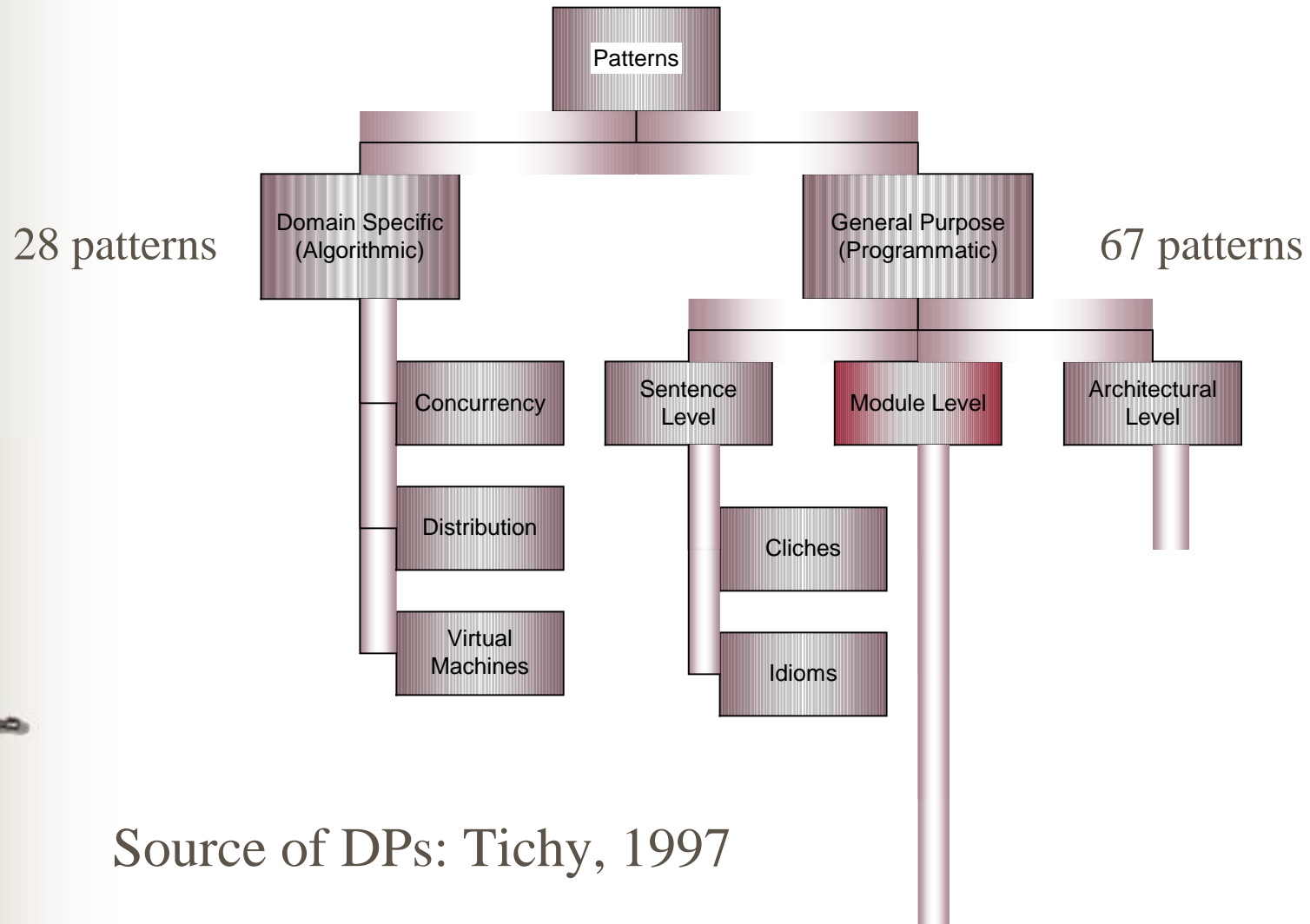
Our Taxonomy of Patterns



Source of DPs: Tichy, 1997



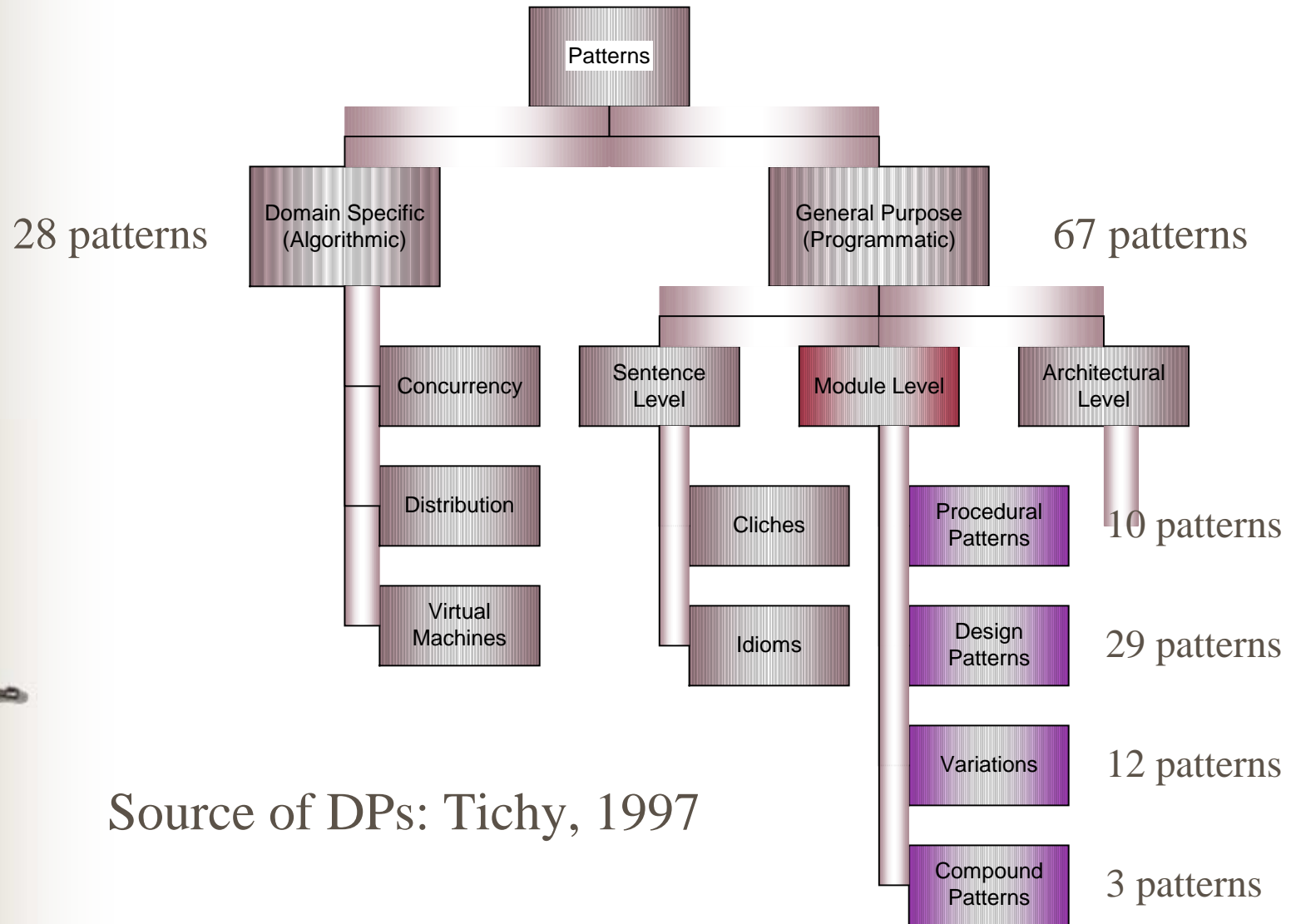
Our Taxonomy of Patterns



Source of DPs: Tichy, 1997



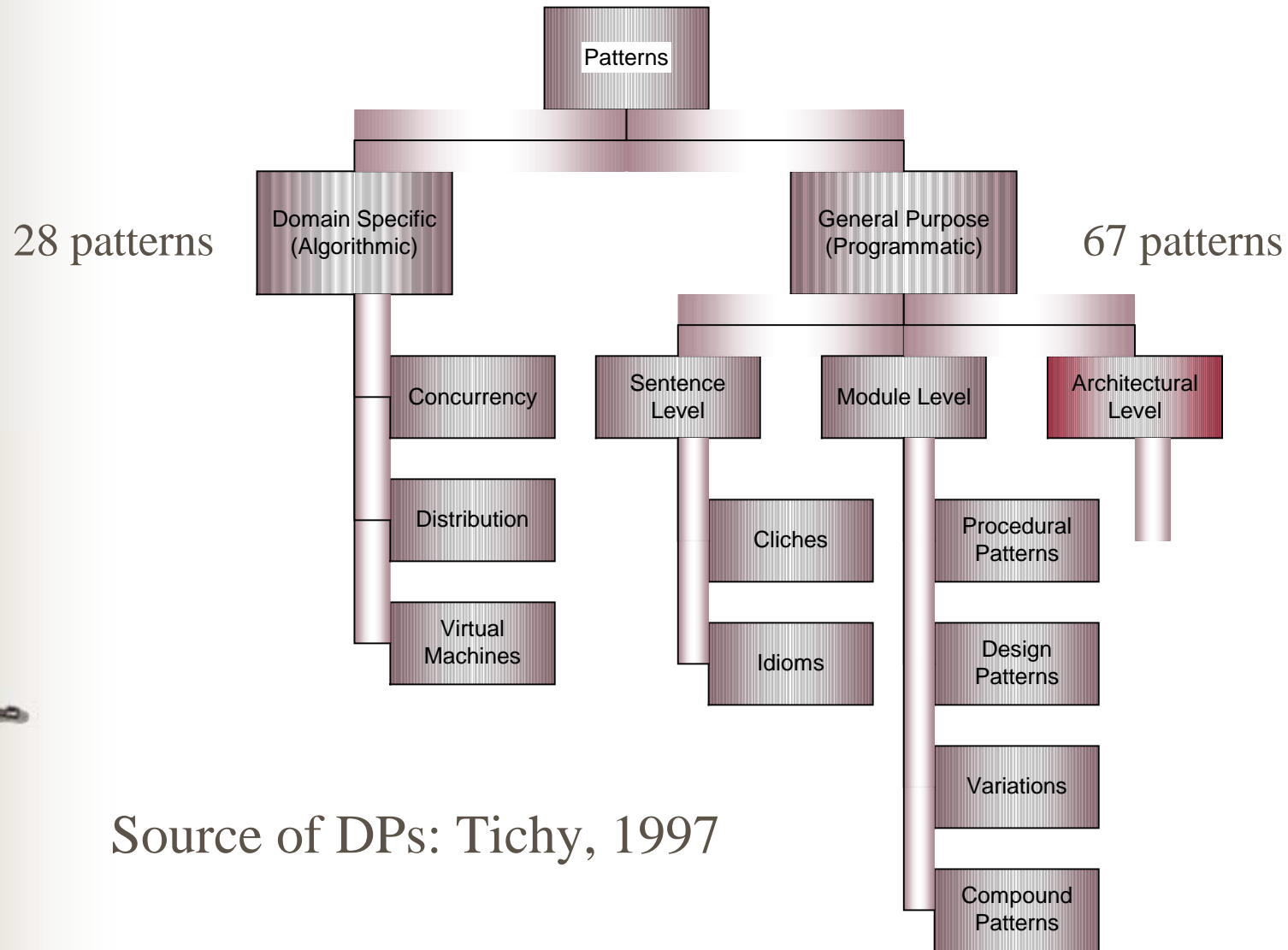
Our Taxonomy of Patterns



Source of DPs: Tichy, 1997



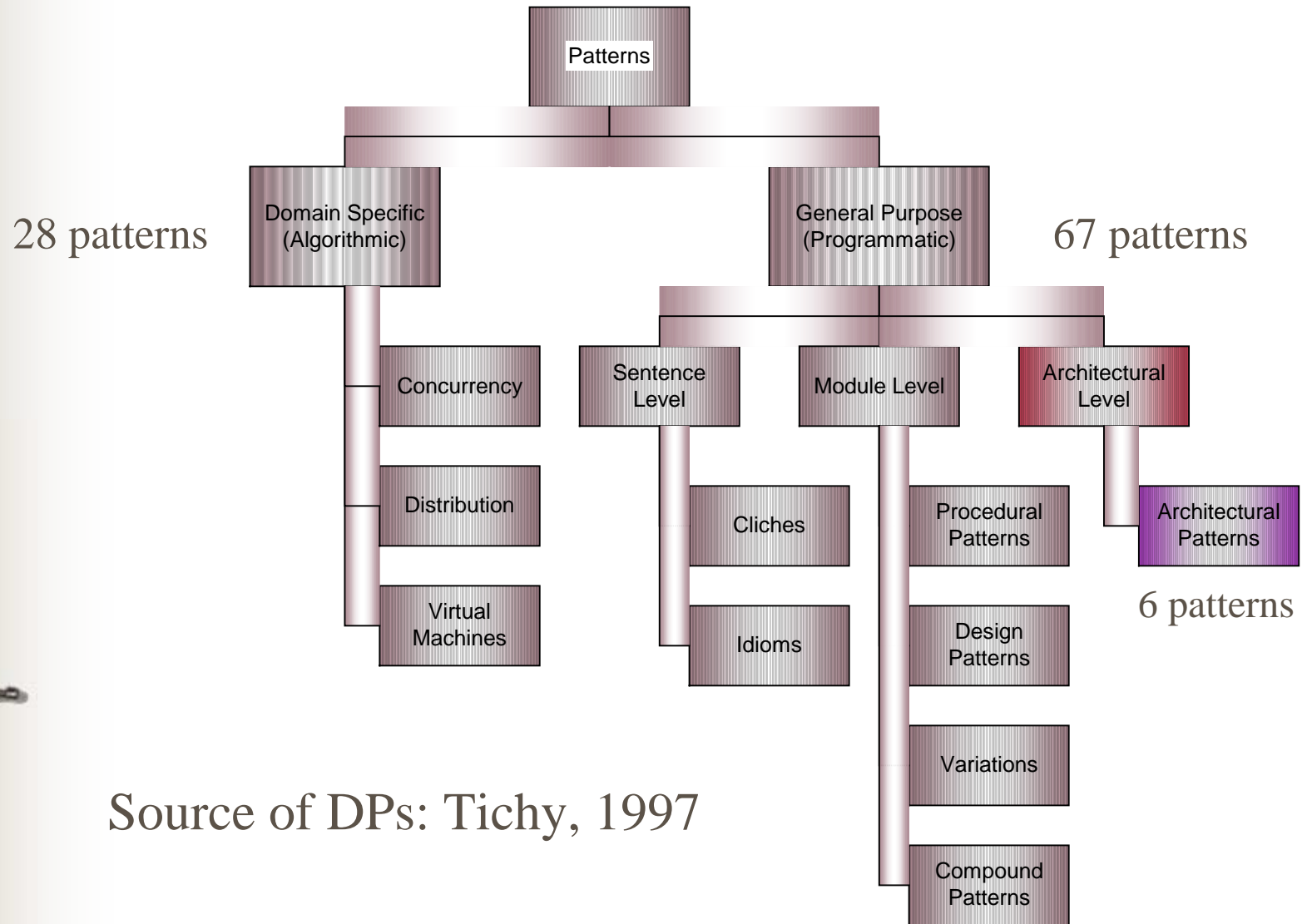
Our Taxonomy of Patterns



Source of DPs: Tichy, 1997



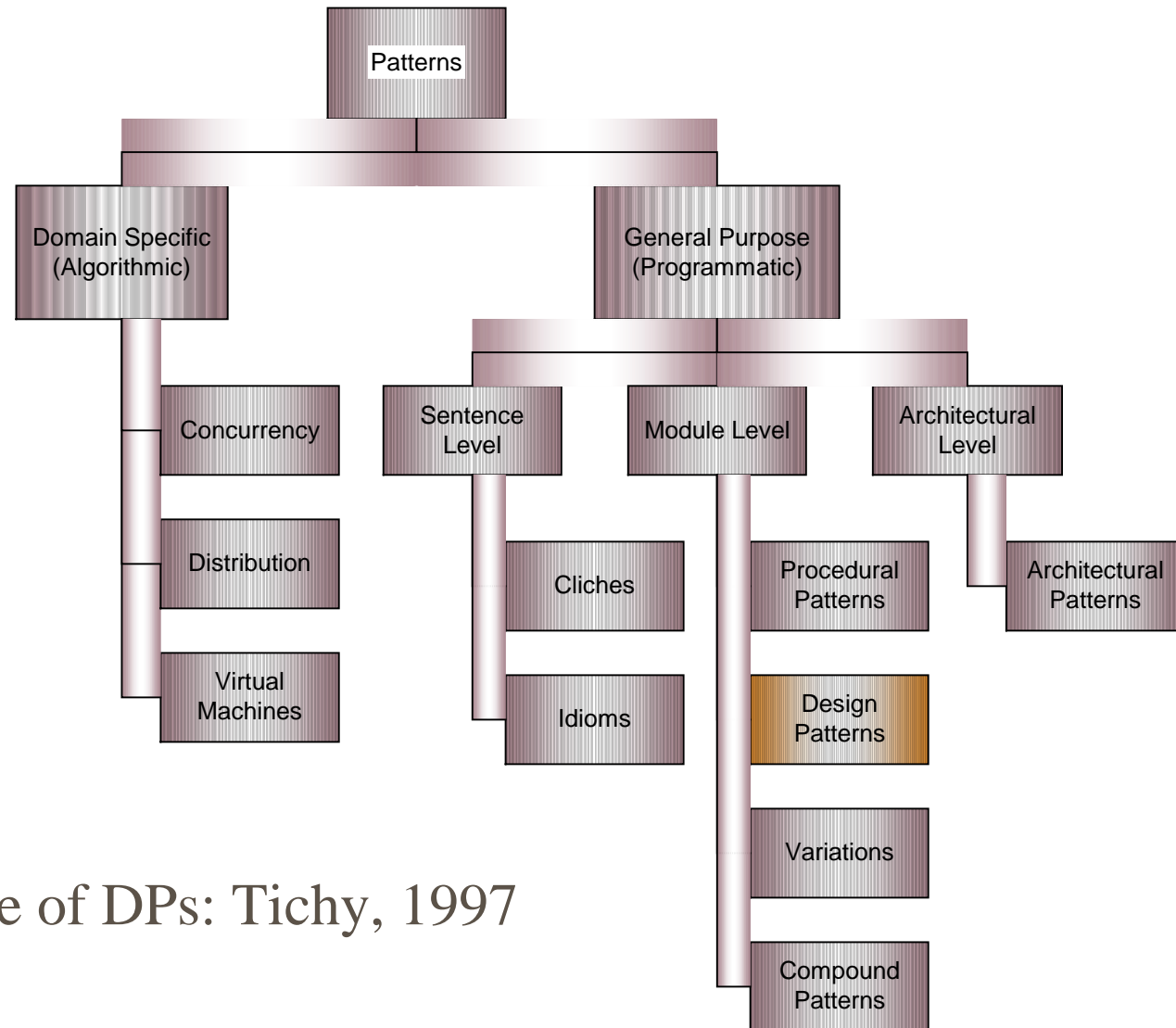
Our Taxonomy of Patterns



Source of DPs: Tichy, 1997



Design Patterns Data Set



Source of DPs: Tichy, 1997





Design Patterns in the Data Set

■ GoF patterns

■ Creational

Abstract Factory	Prototype
Builder	Singleton
Factory Method	

■ Structural

Adapter	Extension Objects
Bridge	Facade
Composite	Flyweight
Decorator	Proxy

■ Behavioral

Chain of Responsibility	Memento
Command	Observer
Interpreter	State
Iterator	Strategy
Mediator	Visitor

■ Additional patterns

■ Extension Objects
Gamma, 1998

■ Manager
Sommerlad, 1998

■ Pipeline
Vermeulen, Beged-Dov and
Thompson, 1995

■ Product Trader
Baumer and Riehle, 1998

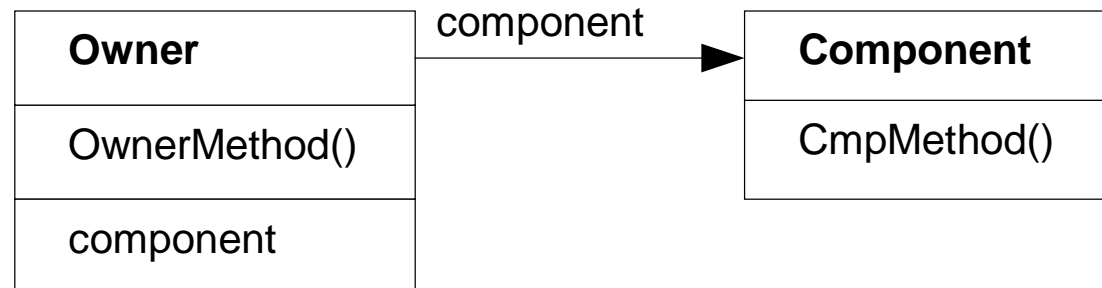
■ Sponsor-Selector
Wallingford, 1998

■ Type Object
Johnson and Woolf, 1998



Composition

Structure of the composition





Composition vs. Inheritance

Inheritance

- Big class hierarchies.
- Break of encapsulation.
- Binds at compile time.

Composition


- Hard to understand.
- Not enough existing components.
- Binds at run-time.

Composition and inheritance should be used in combination.





Composition in Design Patterns

- 
- Inheritance: Adapter (class)
 - Abstract base class: Factory Method
Template Method
 - No inheritance: Singleton
 - Composition: [rest of the DPs]
 - Interfaces: [most DPs]



Implementing Composition

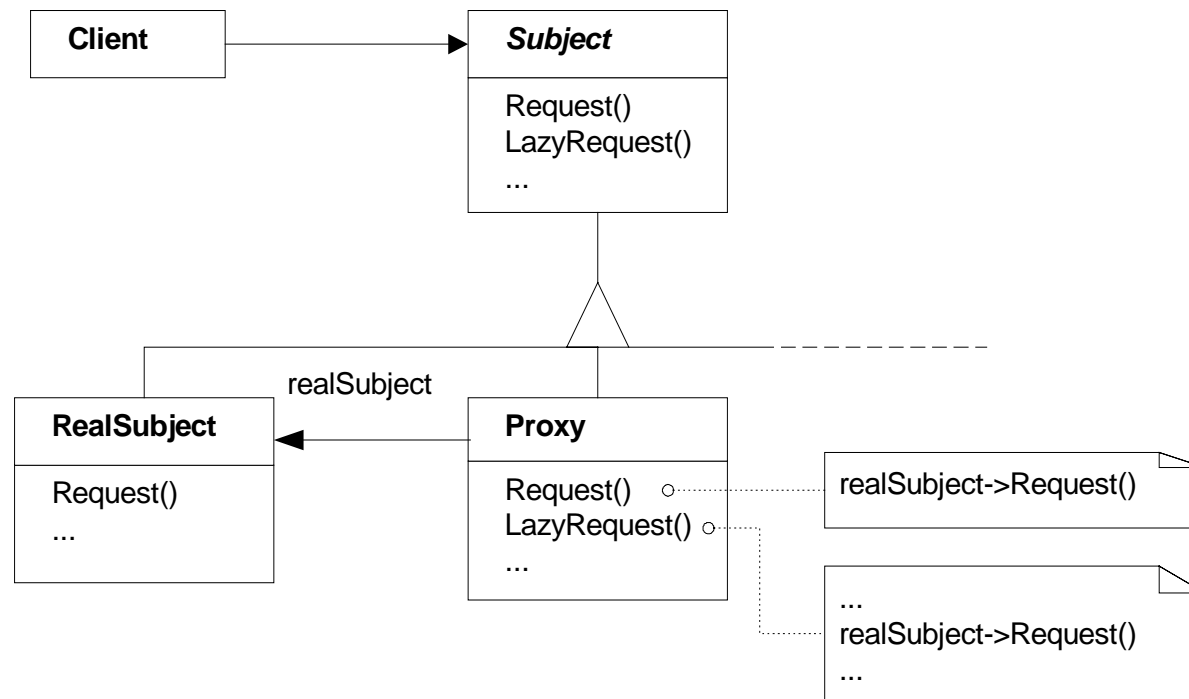
Implementation Criteria Set

- Coupling strength
- Time of coupling
- Features export
- Overriding prior to export
- Internal dynamic binding
- External dynamic binding

Criteria suggested by Gil and Lorenz, 1997.



Example – PROXY pattern



- Purpose: placeholder for another object.



Coupling Strength

“How strong is the connection between the participants of the composition?”



Attribute

- *reference or value ?*
- *references or pointers ?*
- *exclusive or shared ?*
- *single or list of objects ?*

PROXY pattern

- *reference semantics.*
- *use of pointers.*
- *shared component.*
- *single component.*

Exporting and Overriding Features

“Is the connection exported?”


“May the connection be overridden before it is exported?”

PROXY pattern:

Features		Overriding		
Name	Number	none	replace	<i>α</i> -refine
Request ()	group	√		
LazyRequest ()	group		√	√



Conclusions

- Gained insight of:
 - Syntactic mechanisms found in design patterns.
 - Language features useful to implement design patterns.
 - Future of mainstream OOP languages.
- 



Proposed Lingual Construct

■ *Connectors*

Ducasse and Richner, 1997 –

FLO model in CLOS and SMALLTALK.

■ Additional Requirements:

- Type checking of components
- Method refinement
- Substitution of the component

